Subject: Re: 2024rc1
Posted by Lance on Mon, 21 Oct 2024 10:55:22 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 21 October 2024 03:05
How? I have noticed that some people tend to constexpr to everything, but I fail to see a reason. If that is supposed to perform the test only in compile time, then 30 years old compiler will do that anyway. But I might be missing something perhaps?

On a second thought, you are right. A reasonably good compiler would perform the optimization anyways. I remeber a couple years ago when I dig into u++ memory allocation utilities, I saw this one

```
template <class T>
void memcpy_t(void *t, const T *s, size_t count)
{
 if((sizeof(T) & 15) == 0)
  memcpy128(t, s, count * (sizeof(T) >> 4));
 else
 if((sizeof(T) & 7) == 0)
  memcpy64(t, s, count * (sizeof(T) >> 3));
 else
 if((sizeof(T) & 3) == 0)
  memcpy32(t, s, count * (sizeof(T) >> 2));
 else
 if((sizeof(T) & 1) == 0)
  memcpy16(t, s, count * (sizeof(T) >> 1));
 else
  memcpy8(t, s, count * sizeof(T));
}
```

Now I know you already counted on the compile time optimization.

Then the question becomes: what constexpr-if has to offer? Maybe speak a programmer's intention more explicitly and thus possible compiler check, like override?